数据库的特点:实现数据共享,减少数据冗余:采用特定的数据类型:具有较高的数据独立性,具有统一的数据控制功能｡

行记录,字段

主键(primary key)用于唯一的标识表中的每一天记录｡可以定义表中的一列或者多列为主键,主键列上不能有两行相同的值,也不能为空值｡

**创建数据库**:create database 数据库名

**查看数据库:**show databases;

**使用数据库:**use 库名

**建表:**create table 表名 (字段名 数据类型);

**修改结束符:**delimiter 结束符

**查看表**:show tables;

**删除数据库**:drop database 数据库名(生产环境谨慎操作)

一旦删除数据库,在库里面所有的表所有的数据将随着数据库的删除一并删除,谨慎操作

**查看版本:**SELECT VERSION()

**写入数据:**insert into 表名 values (数据)

**单个表中主键唯一**

**mysql中是否允许出现同样的库 不可以**

**一个库中是否允许出现同名的表 不可以**

**一个表中是否允许出现同名的字段 不可以**

**区分大小写:库名,表名**

**不区分大小写:字段名**

**CREATE TABLE haha**

**-> (**

**-> student\_id INT UNSIGNED,**

**-> name VARCHAR (30),**

**-> sex CHAR(1),**

**-> birth DATE,**

**-> PRIMARY KEY (student\_id));**

**INSERT INTO haha VALUES (1,'user1','m','1998-10-22');**

**查看数据:**SELECT \* FROM 表名

**查看数据库版本:**SELECT VERSION() ;

**数据库技术构成**

数据库系统由硬件部分和软件部分共同沟通,硬件主要用于存储数据库中的数据,包括计算机,存储设备等,软件部分侧主要包括DBMS､支持DBMS运行的操作系统,以及支持多种数据进行应用开发的访问技术等

**数据库系统**

数据库系统有三个主要的组成部分

数据库:用于存储数据的地方

数据库管理系统:用于管理数据库的软件

数据库应用程序:为了提高数据库系统的处理能力所使用的管理数据库的软件补充｡

**SQL语言**

在92年由美国ANSi(国家标准机构)发布｡sql-92 在99年被优化过,叫做sql-99sql语言为4大类

**数据定义语言DDL**(data fefinition language):DROP(删除)､ALTER(修改)､CREATE(创建) 对于库和表而言

**数据操作语言DML**(data manipulation language):INSERT(写入)､UPDATE(更新)､DELETE(删除) 对于数据而言

**数据查询语言DQL**(data query language):SELECT(查询)

**数据控制语言DCL** (date control language):BEGIN(开始)､ROLLBACK(回滚)､COMMIT(提交)､GRANT(授权)､REVOKE(撤销) 对于用户或者功能的控制

**访问技术**(ODBC,JDBC,ADO.NET,PDO)

**mysql优点**(规模小,功能有限,但是体积小,速度快､成本低,且它提供的功能对稍微复杂的应用来说已经够用的)

**mysql优势**(速度快,开源-对于绝大部分人来说是免费的,上手简单,容易使用,移植性强,接口丰富,安全性和连接性)

**数据目录**

/usr/local/mysql/data/

test:属于测试库,所有用户对于该库都拥有root权限(不会存放重要信息)

mysql:用于存放用户权限,密码信息和关键字等控制和管理信息

information\_schema(死key吗):存放mysql服务器所维护的其他数据库的信息,比如数据库名,表名,约束条件访问权限等

performance(破佛们死)\_schema:5.5版本之后出现,记录mysql服务器性能参数

**数据库存储引擎**

位于数据库底层,属于底层组件,提供增删改查的数据操作｡不同的存储引擎有不同的功能,属于mysql核心

**查看当前的mysql的所有存储引擎**

show engines(安真死)\G

Engine: 引擎名字

Support(死剖特): mysql是否支持当前引擎

Comment: mysql对该存储引擎的描述

Transactions(chu啊森死): 是否支持事务

XA: 是否支持事务分布式

Savepoints(贼伍炮昂死): 是否支持保存点

merge(吗之) : 属于MylSAM组合,属于MylSAM聚合器,merge表中没有数据｡

archive(啊开五): 在5.5版本之后开始支持索引,属于压缩表｡仅支持写入数据和查询数据两种功能

csv 将数据文件变为csv格式进行存储,不支持索引,合适用于数据交换

mrg\_myisam 属于合并的MylSAM表｡

SHOW VARIABLES(外儿饿boos) LIKE 'default\_storage\_engine'; 查看默认存储引擎

SHOW VARIABLES LIKE '参数'; 查看系统参数功能

存储引擎

不同的存储引擎提供不同功能,占用的空间大小不同,读取性能不同

在mysql中每一个表都对应一个存储引擎,默认innodb

**MyISAM(买isam)**

1.属于5.5版本之前的默认存储引擎

2.读取速度快,占用资源少,但是不支持事务,不支持外键,支持全文索引,空间索引

3.读写相互阻塞

4.只能缓存索引不能缓存数据

**MyISAM应用场景**

1.不适合事务的业务

2.读取较多的业务

3.并发量低的业务,且数据修改较少的业务

4.硬件资源受带限制的情可以考虑使用MyISAM

**lnnoDB**

1.使用事务首选存储引擎,5.5版本之后作为默认存储引擎

2.具有提交,回滚,崩溃恢复能力的事务安全存储引擎,可以处理大并发量,性能较高,支持外键

3.缓存性能高,可以缓存数据也可以缓存索引,但是对硬件要求较高

4,一旦使用innodb,会在数据目录下产生两个名字为lib\_logfile0 和ib\_logfile1的日志文件,ibdata1和10M大小自动扩展的文件

ibdata1:属于二进制文件不能通过文本编辑器进行打开,且使用innodb存储引擎的表,表中的数据和索引都会存放在该文件中

**InnoDB应用场景**

1.需要事务的业务,例如银行转账

2.更新数据较为频繁的业务,比如微博

3.数据的一致性要求较高的业务,比如充值转账

**Memory**

1.将所有的数据保存到内存中,提供用户的快速访问(易失性)

2.memory如果不需要该存储引擎中数据,直接将表清空或删除,可以直接释放内存

3.memory支持hash和btree索引(算法),不支持设置blob和text数据类型

**关于表的操作**

一个表一定对应一个存储引擎,一个字段一定对应一个数据类型

语法:

create table 表名(

字段1 数据类型[完整性约束条件]

字段2 数据类型[完整性约束条件]

);

约束条件

对于字段进行限制,要求用户写入符合要求的数据,如果写入的数据不满足约束条件的要求,将不会写入

约束条件

PRIMARY(拍摸瑞) KEY 标识该属性为该表的主键,可以唯一的标识对应数据

FOREIGN (佛润)KEY 标识该属性为该表的外键,是与之联系的某表的主键

NOT NULL(闹特NO) 标识该属性不能为空

UNIQUE (有你可) 标识该属性值唯一

DEFAULT 为该属性设置默认值

AUTO\_INCREMENT(in扣们特) 设置该属性为自动增加

**主键**

**1.唯一**

**2.非空**

**语法**

**1.字段名 数据类型 PRIMARY**(拍摸瑞) **KEY**

例子:

CREATE TABLE hehe (id INT PRIMARY KEY, name VARCHAR(30), sex BIT);

**2.在定义完所有的列之后添加主键**

**PRIMARY**(拍摸瑞) **KEY (添加主键的字段)**

例子:

CREATE TABLE hehe1 (id INT, name VARCHAR(30), birth DATE, d\_no INT, sakary DECIMAL, PRIMARY KEY(id) CREATE TABLE hehe2 (name VARCHAR(30), sex CHAR (1), address VARCHAR (30), birthday DATE, PRIMARY KEY(name,sex,address,birthday)););

**多字段的联合主键**

PRIMARY(拍摸瑞) KEY (字段1,字段2...)

例子

CREATE TABLE hehe2 (name VARCHAR(30), sex CHAR (1), address VARCHAR (30), birthday DATE, PRIMARY KEY(name,sex,address,birthday));

**非空**

表示表中的数据不能写入空值,但是可以写入重复值

语法

字段名 数据类型 NOT NULL

例子:

CREATE TABLE feikong (id INT NOT NULL);

**唯一性约束条件**

设置该属性的字段不允许写入重复值

语法

1.字段名 数据类型 UNIQUE

2.[CONSTRAINT 约束名] UNIQUE (字段名)

**默认值(缺省值)**

如果不忘设置默认值的字段写入数据,mysql会自动将设置好的默认值写入字段内

语法

字段名 数据类型 defaule 默认值

**设置自增约束(数值类数据类型)一个表中只允许有一个自增字段**

用于给表中写入的数据生成新的唯一ID

语法

字段名 数据类型 AUTO\_INCREMENT

SHOW VARIABLES LIKE ‘auto\_increment\_increment’ 查看自增状态

SET auto\_increment\_increment=数值 设置自增量

**查看表结构**

DESC 表名

DESCRIBE 表名

**查看详细表结构**

SHOW CREATE TABLE 表名\G

\G 以更简介更直观的方式显示表中数据

\g=;

**外键**

外键用来建立表与表之间联系的工具(基于字段建立)

父子表,谁建立外键那个表为子表,建立外键的字段只有父表中有的数据字段中才允许写入,子表中不允许写入父表没有的数据(仅建立外键的字段)

语法

[CONSTRAINT 约束名]FOREIGN(佛润) KEY (建立外键的字段) REFERENCES(rva分润死) 父表名(父表主键列)

关联父表时,父表的字段必须为主键

**修改表名**

ALTER TABLE 表名 RENAME [TO] 新名字

如果带有TO的话,可以进行库与库之间表迁移,并且可以在迁移时修改表名

例句

ALTER TABLE 表名 RENAME TO 库名.新名字;

**修改数据类型**

ALTER TABLE 表名 MODIFY(毛的fai) 字段名 数据类型

**修改字段名**

ALTER TABLE 表名 CHANGE(陈只) 旧字段名 新字段名 数据类型

**添加字段**

ALTER TABLE 表名 ADD 字段名 数据类型 [约束条件][FIRST|AFTER(啊弗特) 已经存在的字段]

example2添加新字段名为address数据类型为varchar(20)约束条件为非空,添加到birth字段之前

例子:

ALTER TABLE example2 ADD address VARCHAR (20)NOT NULL AFTER name;

添加到最后

ALTER TABLE example2 ADD a INT ;

**删除字段**

ALTER TABLE 表名 DROP 字段名;

例子

ALTER TABLE example2 DROP ;

**修改字段的排列顺序(已经存在字段)**

ALTER TABLE 表名 MODIFY(毛地fai) 字段名 数据类型 FIRST|AFTER 已经存在的字段

**删除外键约束条件**

ALTER TABLE 表名 DROP FOREIGN KEY 外键的约束

**修改表的存储引擎**

ALTER TABLE 表名 engine=要修改的引擎

忘记表放在那个库中

使用 information\_schema 在该库中又名叫TABLES的表,在该表中有TABLE\_SCHEM字段表示表所在的库名,TABLE\_NAME表示表名,如果忘记在那个库中使用

SELECT TABLE\_SCHEMA FROM TABLES WHERE TABLE\_NAME='表名';

忘记外键在那个表中

在information\_schem库中有名为TABEL\_CONSTRAINTS的表,记录了所有的约束条件(主键,外键,唯一性约束条件)通过该表可以找到指定的约束条件所在的表以及数据库

SELECT \* FROM TABLE\_CONSTRAINTS WHERE CONSTRAINT\_TYPE='FOREIGN KEY'\G

删除主键

ALTER TABLE 表名 DROP PRIMARY(拍摸瑞) KEY

删除表

DROP TABLE [IF EXISTS] 表名

删除唯一性约束条件

ALTER TABLE 表名 DROP KEY 约束名(非字段名) ;

删除非空,自增,默认值

在使用modify或者change修改数据类型时只要不写入上述类型,等同于删除

在已经存在的表上添加,非空,自增,默认值

在已经存在的表上添加主键,外键,唯一性约束条件

主键,唯一性约束条件既可以通过add添加,也可以通过modify,change来添加(两种语法)

非空,自增,默认值只能通过chage或者modify添加

外键只能通过add添加

**数据类型**

数据类型时决定数据写入格式的工具,一旦规定了数据类型就确定了数据的写入方式,比如整数类型只能写入整数,不可以写入字符串或者二进制数据

1.数值类型

整数类 存储需求 取值范围

TINYINT(提呢类特) 取值范围极小的整数 1字节 -128-127

SMALLINT 取值范围小的整数 2.字节 -32768-32767

MEDIUMINT 取值范围中等的整数 3.字节 -8388608-8388607

INT 普通整数 4字节 -2147483648-2147483647

BIGINT 取值范围较大的整数 8.字节

-9223372036854775808-9223372036854775807

BIGINT(BIG尼特)

小数类

浮点类型

定点类型

**无符号取值范围(非负整数)**

1 0-255

2 0-65535

3 0-16777215

4 0-4294967295

5 0-18446744073709551415

字段后面括号中的数值为显示宽度,允许我们写入指定长度范围之内的数据

**小数类**

**浮点数类**

FLOAT (M,N) 单精度浮点数 4字节

DOUBLE (M,N) 双精度浮点数 8字节

M表示一共可以写入的位数,为精度

N表示小数可以存在的位数,为标度

定点数类

DECIMAL (M,N)以字符串形式存放数据,精确度比浮点数据类高,M+2

科学运算货币运算使用定点数,如果取值范围较大,使用浮点法

**设置无符号取值范围**

字段名 数据类型 UNSIGNED

**truncate 和delete 区别**

1. tryncate table 一经写入删除表中所有数据,没有where条件判断

delete from 写入可以通过where条件判断进行删除指定行数操作

2.truncate 比 delete 速度快,tryncate 占用的资源少(日志资源,事务资源)

3.使用truncate删除表数据表的大小和自增数据将会恢复到初始大小,而delete 不能

4.使用truncate 删除会释放表空间,而delete不释放

**2.日期/时间数据类型**

类型 格式 存储需求 取值范围

YEAR YYYY 1字节 1901-2155

TIME HH:MM:SS 3字节 -838:59:59-838:59:59

DATE YYYY-MM-DD 3字节 1000-01-01-9999-1231

DATETIME YYYY-MMM-DD HH:MM:SS 8字节 4222-01-01 00:00:01-9999-12-31 23:59:59

TUMESTAMP YYYY-MMM-DD HH:MM:SS 4字节 1970-01-01 00:00:01-2038-01-19 03:14:07 UTC

**YEAR**

字段 YEAR

数据写入规则

‘0’或者‘00’-‘69‘ 被识别为2000-2069

‘70‘-’99‘ 被识别为1970-1999

01-69 被识别为2001-1999

0 被识别为0000

**TIME数据写入规则**

三组数据正常识别 时分秒

两组数据被识别为 时分

一组数据被识别为 秒

带有空格时,空格前面的数值被识别为天数,空格后有一组数据被识别为小时,两组数据被识别为 时分,三组正常识别 时分秒

**DATE数据写入规则**

当年份以两位数形式写入时

00-69 被识别为2000-2069

70-99被识别为1970-1999

**DATETIME**  
mysql 中支持不严谨的语法,分隔符可以用其他符号进行表示

年份以两位数的形式写入时,详情参照date类型

**TIMESTAMP**

UTC=世界标准时间

设置时区

SET TIME\_ZONE=’时区‘

**3.字符串数据类型**

**1.文本字符串**

类型 存储需求

CHAR (M) 固定长度文本字符串 M字节 M<=255

VARCHAR (M) 可变长度的文本字符串 L+1字节 M<=65535

TINYTEXT 非常小的文本字符串 L+1字节

TEXT 较小的文本字符串 L+2字节

MEDIUMTEXT 中等大小的文本字符串 L+3字节

LONGTEXT 较大的文本字符串 L+4字节

CHAR数据类型中(M)作为显示宽度,允许用户写入M以内的字符数量,如果写入的数据超过显示宽度多余的部分将会被mysql删除,如果写入的数据没有达到指定的显示宽度,mysql会在数据的尾部补充空格达到显示宽度,如果在显示时尾部检测到空格,空格会被删除

varchar(M)varchar数据实际占用空间不需要通过显示宽度确定,只能通过写入数据决定最后的大小,且不会在数据后面添加空格达到执行显示宽度

**TEXT**

1.TINYTEXT 最多存放255字符的数据

2.TEXT 最多存放65535字符的数据

3.MEDIUMTEXT 最多存放16777215字符的数据  
4.LONGTEXT 最多存放4294967295字符数据

存放数据库选择合适的数据对于空间来说,可以达到资源利用最大化｡

**ENUM(枚举)**

在给定的范围内选择一个值作为结果

语法

字段名 ENUM(‘v1’,‘v2’,...’vn’)

每个写在枚举中的数据叫做元素,每个元素都有一个索引下标,从1开始,以此类推,写入数据时可以通过索引下标的方式写入数据,也可以通过写入给定范围值的方式写入数据,不可以向枚举中添加不存在范围内的元素,否则报错

**SET**

在给定的范围中选择0个或者多个值作为结果

语法

字段名 se(‘v1’,’v2’...’vn’)

和枚举不同的是,set中可以选择多个值作为结果,而枚举中只能选择一个,和枚举相同的时,set中的元素也可以索引下标,如果写入到set中的值元素有重复,set将会给重复值进行删除

**2.二进制字符串(图片,视频,音频等)**

类型 存储需求

BIT 位字段数据类型 约为(M+7)/8字节

BINARY 固定长度的二进制字符串 M字节

VARBINARY 可变长度的二进制字符串 L+1字节

TINYBLOB 非常小的二进制对象 L+1字节

BLOB 小的二进制对象 L+2字节

MEDIUMBLOB 中等大小的二进制对象 L+3字节

LONGBLOB 较大的二进制对象 L+4字节

BIT作为字段数据类型,存放数据时使用二进制进行存储,显示宽度的数据表示对二进制字符串的限制,不是对数据写入数据的字符长度进行限制｡

BIN函数将二进制数据显示出来

BINARY(M)如果写入的数据没有达到指定的显示宽度,将会在数据的末尾填充\0达到指显示宽度

VARBINARY(M)如果写入的数据没有达到执行显示宽度,不做任何操作

LENGTH 计算字节长度的函数

BLOB存储二进制字符串,TEXT存放文本字符串

BLOB没有字符串,TEXT有字符集

数据类型 存储空间

TINYBLOB 最大存放255字节

BLOB 最大存放65535字节

MEDIUMBLOB 最大存放16777215字节

LONGBLOB 最大存放4294967295字节

CHAR和VARCHAR在读取性能看,char的速度要快于varchar

ENMU和SET

ENUM和SET 如果选择有多个,使用SET,如果单项选择使用ENUM

BLOB和TEXT 如果存放图片,视频等用BLOB､存放大量文本使用TEXTs

TINYBLOB

**运算符**

1.算数运算符

+､-､\*､/､%

2.比较运算符

= 等于

<=> 安全等于

> 大于

< 小于

>= 大于等于

<= 小于等于

<>,!= 不等于

IS NULL 判断一个值是否为空值

ISNULL 同理

IS NOT NULL 判断一个值是否不为空值

BETWEEN AND 判断一个值是否介于两个值之间

LEAST 多个数值最小值

GREATEST 多个数值最小值

IN 判断一个值是否为IN列表中的值  
LIKE 通配符匹配  
REGEXP 正则表达式匹配

NOT IN 判断一个值是否不是IN列表中

等于,小于,大于...没有办发判断空值

安全等于可以判断空值

BETWEEN 判定规则,被判定的数据需要大于等于and前的数据,小于等于and后的数据,才被判定介于两者之间否则为假命题

like 通配符查询

% 匹配任意字符

\_ 匹配任意单个字符

^ 匹配以什么开头的数据

$ 匹配以什么结尾的数据

. 匹配任意单个字符

[] 匹配方括号任意字符

\* 匹配星号前面的字符零次或者多次

| 匹配管道符前面任意字符串

3.

逻辑非 NOT或者!

1.当操作数为0时,返回1

2.当操作数不为0时,返回0

3.操作数为NULL时返回NULL

逻辑与 AND或者&&

1.当所有数不为0,且均不为NULL时,返回1

2.当有一个或者多个操作为0返回0

3.其余情况返回NULL

逻辑或 OR或者||

1.当两个操作数不为Null,且任意 一个不为0时,返回1,否则为0

2.当有一个为NULL,另一个操作不为0,返回1,否则为NULL  
3.当两个操作数为NULL时,返回NULL

逻辑异或 XOR

a XOR b == ((NOT a) AND b) OR (a AND (NOT b));

1 XOR b == ((NOT 1) AND 0 ) OR (1 AND (NOT0 ))

1 当任意一个操作为NULL,返回NULL

2当所有操作不为空,两个操作数都为0或者都不为0,返回0

3.如果一个为0,另一个不为0,返回1

4.位操作运算符

位或 |

对应二进制位有一个位1,则为1,否则为0

位与 &

对应的二进制位都为1,则为1,否则为0

^ 位异或

对应的二进制位相同返回0,不同返回1

<< 位左移

将指定的二进制位向左移动指定的位数,左移之后,左边高位将被移除并丢失,右边地位用0补齐

>> 位右移

将指定的二进制位向右移动指定的位数,右移之后,右边高位将被移除并丢失,左边地位用0补齐

~ 位取反

将对应的二进制位逐位翻转,即0边1,1变0

运算符优先级:由高到低

=

||or

XOR

&& AND  
NOT

BETWEEN

=,⬄,<,<>,!=,IS NULL,IS NOT,LIKE,IN,REGEXP

|

&

^

<<,>>

-+

\*/%

·~

**函数**

1.ABS(x) 平方根函数 不能为负数

2.PI() 圆周率函数

3. SQRT() 平方根函数 不能为负数

4. MOD(x,y) 求余函数 x为被除数,y为除数,之后取余数

5.CEIL(x) 取整函数 取最近接近X且不小于X的整数

CEILING(x) 取整函数 同上

FLOOR(x) 取整函数 取最接近x且不大于x的整数

6 RAND() 随机函数 获取一个随机数,范围介于0-1之间(不等于0且不等于1)

RAND(X) 随机函数 x作为种子值,用于获取重复的随机数列

7.ROUND(X) 四舍五入函数x作为目标进行四舍五入操作

ROUND(X,Y) 四舍五入函数y作为保留的小数函数

8.TRUNCATE(x,y) 截取数值的函数

9.SIGN(x) 符号函数对于x进行判断,大于0返回1,小于0返回-1,等于0返回0

AS别名 给表头起一个新的名字

10. CHAR\_LENGTH(x) 计算字符串长度函数统计字符个数

LENGTH(x) 计算字符串长度函数计算字符长度

在字符集是UTF8的情况下,一个汉字占用一个字符长度,如果字符集为不支持中文的例如:latin1一个汉字占用3个字符长度,汉字占用3个字节空间

11. CONCAT(s1,s2,..)用于合并字符串s1,s2... 合并字符串的长度如果元素中出现NULL则返回NULL

CONCAT\_WS(x,s1,s2...) 合并字符串的长度如果空值作为元素,mysql忽略,如果作为分隔符返回null

12.s=string字符串 替换字符串函数

INSERT(s1,x,len,s2) 从s1的第x位使用s2替换len长度

13. LEFT(s,n) 获取指定长度字符串函数用于返回字符串s左边的n个字符

RIGHT(s,n) 获取指定长度字符串函数用于返回字符串s右边的n个字符

14. LPAD(s1,len,s2) 填充字符串函数给s1左侧进行填充,使用s2填充到len长度

RPAD(s1,len,s2) 填充字符串函数给s1右侧进行填充,受用s2填充到len长度

15.LTRIM删除左侧空格 删除空格的函数

RTRIM 删除右侧空格 删除空格的函数

TRIM 删除两侧的空格 删除空格的函数 删除空格时删除到第一个不为空的数据为止

16.TRIN(s1 from s) 删除指定字符串的函数 删除字符串s两边的s1,删除s1直到两边检测到不为s1的字符串为止

17.重复字符串的函数

REPEAT(s,n)将字符串s重复生成N边

18.生成空格的函数

SPACE (n)指定生成n个函数

19替换函数

REPLACE (s,s1,s2) 将s中所有的s1使用s2进行替换

20.比较字符串大小的函数

STRCMP (s1,s2)

如果s1大,返回1,如果s2,大,返回-1,如果一样大返回0

SELECT COLLATION(‘abc’);查看排序规则的函数,在utf8的排列规则中不需区分大小写

21获取子字符串的函数 s=string

SUBSTRING (s,n,len)n表示从第几位开始获取,len表示获取的位数

MID(s,n,len)同上如果n位负数,将会倒向获取指定len的字符串

22.翻转字符串函数

REVERSE(s)及首变尾,尾变首｡

23.返回指定位置字符串的函数

ELT(n,s1,s2...)

24.返回指定字符串的位置

FIELD(s,s1,s2,s3...)返回s后面s1-sn的位置

25.返回子字符串位置的函数

FIND\_IN\_SET(s1,s2)返回s1在s2字符串列表中的位置

26.获取当前时间的函数

CURDATE()

CRRENT\_DATE()

27.获取当前日期的函数

CURTIME()

CURRENT\_TIME()

28.获取当前日期和时间的函数

CURRENT\_TIMESTAMP()  
LOCALTIME()

NOW()  
SYSDATE()

29.获取时间戳的函数

UNIX\_TIMESTAMP()

30将时间戳转换为时间的函数

FROM\_UNIXTIME()

31.获取UTC日期的

UTC\_DATE()

YTC\_TIME()

32.获取月份的函数

MONTH 返回对应的月份序号

MONTHNAME 返回对应的月份英文名

33.获取星期的函数

DAYNAME(date)返回日期对应的工作日的英文名

DAYOFWEEK(date)返回日期对应的一周中的索引(1为周天以此类推)

WEEKDAY(date)返回日期对用的工作日的索引(周一为0,以此类推)

WEEK(date)判断当前日期是一年中的第几周(以0开始)

WEEKOFYEAR(date)判断当前日期是一年中的第几周(以1开始)

DAYOFYEAR(date)判断当天是所在一年中的第几天

34.获取对应日期的年份

YEAR 获取对应日期的年份

35.获取季度的函数

QUARTER(date)

36.获取小时的函数HOUR()

获取分钟的函数MINUTE()

获取秒钟的含糊SECOND()

37.获取日期指定值的函数

EXTEACT(type FROM date)

38.时间和秒钟湘湖转的函数

TIME\_TO\_SEC(time)将和时间转换为秒钟

SEC\_TO\_TIME(time)将秒钟转换为时间

39.计算日期和时间的函数

DATE\_ADD(date,INTERVAL expr type) 对日期进行加运算

ADDDATE(同上)同上  
SUBDATE(date,INTERVAL,expr type) 对日期进行减运算

DATE\_SUB()同上

ADDTIME(date,expr)对于时间进行加运算

SUBTIME(date,expr)对于时间进行减运算  
DATEDIFF(date,date)计算连个日期之间的间隔

datediff计算date1减法date2的值,如果date1大表示距离的天数,如果date2大表示过去的天数

40.将日期和时间进行格式化的函数

DATE\_FORMAT(date,format)将日期进行格式化

TIME\_FORMAT(time,format)将时间进行格式化

GET\_FORMAT(value\_type,format type)将制定类型的数据,格式化成想要形式的字符串

%W工作日对应的英文名

%M月份对应的英文名  
%Y四位数形式显示年份

%H以两位数显示24小时时间

%k如果不满10小时,使用一位数显示24小时制时间  
%I以两位数显示12小时时间

%U表示周数,且周日为每周的第一天

%u表示周数,且周一为每周的第一天

%w一周的某一天,返回序列号,0表示周日以此类推

%m返回月份对对应的序号

41IF(expr,v1,v2),如果expr的运算结果为true返回v1,否则返回v2

42.IFNULL(v1,v2)对于v1进行判断,如果v1为空值,返回v2,如果v1不为空值,返回v1

43.CASE

CASE expr WHEN v1 THEN r1[WHEN v2 THEN r2]...[ELSE rn] END

44.存储过程procedure,嵌入到mysql饿一段程序,通过指定的命令进行触发call

CREATE PROCEDURE test\_pro(IN A int,IN M int)

BEGIN

DECLARE n INT DEFAULT 5; 设置一个变量一开始为5

WHILE n>0 Do SELECT n ; while无限循环

SET n=n-1; 新变量是N-1

END WHILE;E 结束无限循环

END //

45.获取当前的版本号

VERSION

46.获取当前的用户连接数ID

CONNECTION\_ID ()

47.查看用户连接信息的函数

SHOW PROCESSLIST

KILL 用户ID可以强制踢出用户

48 查看当前数据库的函数

database()

schema()

49 查看当前登录用户的函数

USER()

CURRENT\_USER()

SYSTEM\_USER()

50 查看指定字符串字符集的函数

CHARSET(str)

51 查看排序规则的函数

COLLATION(str)

52 加密解密函数

(1)password(str) 将明文密码加密成哈希进行输出(不可逆)

(2)md5(str) 将字符串加密成32位十六进制的字符串(不可逆)

(3)encode(str,pswd\_str) 使用pswd\_str作为密码,加密str,二进制数据

mysql> SELECT PASSWORD('123.com');

mysql> SELECT MD5('123.com');

mysql> SELECT ENCODE('tfhz','hello');

解密函数

DECODE(cry\_str,pswd\_str) 使用pswd\_str作为密码解密前面加密的crypt\_str数据

例:mysql> SELECT DECODE(ENCODE('tfhz','hello'),'hello');

53 格式化函数

FORMAT(x,n)

将x进行格式化,以四舍五入的形式保留小数,并且以字符串形式返回

54 不同数字进行进制转换的函数

CONV(要转换的数值,该数值的进制,转换成的进制)

55 ip地址和数值互相转换的函数

INET\_ATON 将ip地址转换为数值

INET\_NTOA 将数值抓换位ip地址

56.加锁解锁函数

GET\_LOCK(str,timeout)

使用字符串str获取的一个锁,超时时间为timeout秒

1.如果成功得到锁,返回1

2.如果超时,返回0

3.如果事务,返回NULL

RELEASE\_LOCK(str)

将命令str的锁解开

1.如果锁被解开,返回1

2.如果该进程没有创建锁,返回0

3.如果锁不存在返回NULL

注:如果锁被解开,不存在

IS\_FREE\_LOCK(str)

检测名为str的锁是否可用

1.如果可用返回1

2.如果被使用返回0

3.如果参数错误,返回NULL

IS\_USED\_LOCK(str)

检测名为str的锁是否被使用

如果被使用返回改用户的连接标识符,如果没有返回NULL

57.重复执行指定操作的函数

BENCHMARK(count,expr) 将expr重复执行count遍,然后测试服务器性能

58.改变字符集的函数

CONVERT (...USING...) 将字符集转换为另外一种格式显示

59.改变数据类型的函数

CAST (x,AS type)将x在type类型中显示

CONVERT(x,type)同上

**查询**

查询,

语法

SELECT \* |字段 FROM 表1[,表2][WHERE 条件判断][GROUP BY字段][having条件判断][order by 字段][limit offset count]

WHERE 中通常受用比较运算符进行数据的筛选

GROUP BY 分组 去重,适用于重复值较多的数据

ORDER BY 排序 数据排列,适用于重复较少的数据

LIMIT offset count 限制输出条目数

查看表中水果价格为11.2的水果记录

SELECT \* FROM fruits WHERE f\_price=11.2 ;

查看水果表中价格在7-15之间的水果,显示水果名和价格

SELECT f\_name,f\_price FROM fruits WHERE f\_price BETWEEN 7 AND 15;

SELECT f\_name,f\_price FROM fruits WHERE f\_price>=7 AND f\_price<=15 ;

查看水果表中供应商编号为101或者102的记录

SELECT \* FROM fruits WHERE s\_id=101 or s\_id=102 ;

SELECT \* FROM fruits WHERE s\_id IN(101,102) ;

查看水果表中的价格不在5到10之间的水果信息

SELECT \* FROM fruits WHERE f\_price<=5 or f\_price>=10 ;

查看水果表名字以a开头的数据记录

SELECT \* FROM fruits WHERE f\_name LIKE 'a%' ;

查看水果表中名字带有or的数据记录

SELECT \* FROM fruits WHERE f\_name LIKE '%or%';

查看水果表中名字含有a,o,m的数据记录

SELECT \* FROM fruits WHERE f\_name REGEXP '[a,o,m]';

查看水果表中水果以b或者d开头价格高于8的记录

SELECT \* FROM fruits WHERE f\_name REGEXP '[d%,b%]' AND f\_price>8;

SELECT \* FROM fruits WHERE f\_name REGEXP '^[bd]' AND f\_price>8;

运算符

=

>=

<=

<>

>

<

BETWEEN AND 介于两者之间

BETWEEN AND 不在两者之间

IN 位于in列表中

NOT IN 不在in列表中

LIKE 模糊查询

REGEXP 正则表达式查询

IS NULL 判断是否为空

IS NOT NULL 判断是否不为空

AND 并且,和

OR 或者

% 匹配任意字符

\_ 匹配任意单个字符

^ 匹配以什么开头的数据

$ 匹配以什么结尾的数据

. 匹配任意单个字符

[] 匹配方括号任意字符

\* 匹配星号前面的字符零次或者多次

| 匹配管道符前面任意字符串

匹配

order by 字段[,字段] 将指定字段的数据进行排序,可以按照升序降序方式进行排列,默认升序

ASC 升序

DESC降序

GROUP BY 分组 将相同的数据合并到一起形成一条新的记录,去重HAVING通常和GROUP BY连用,表示如果分组之后还有条件写入使用HAVING继续条件判断

HAVING可以使用聚合函数 where不能

SUM求和

AVG求平均值

MIN求最小值

MAX求最大值

COUNT计数

GROUP\_CONCAT 将分组过后的数据显示完整

查看提供水果类大于两种的供应商,显示该供应商提供的水果,以及供应商的编号

SELECT s\_id,GROUP\_CONCAT(f\_name)AS f\_name FROM fruits GROUP BY s\_id HAVING COUNT(\*)>2;

LIMIT 限制输出的结果数量

LIMIT [offset] count

offset 表示偏移量表示从第几条开始输出

count 表示数量 代表输出的条目数量

查看表中价格最高的水果记录

SELECT f\_name,f\_price FROM fruits ORDER BY f\_price DESC LIMIT 1 ;

查看每个供应商提供的价格最高的水果,显示水果名,和供应商编号,以及水果价格

SELECT s\_id,GROUP\_CONCAT(f\_name)name,f\_price FROM fruits WHERE (s\_id,f\_price) IN (SELECT s\_id,MAX(f\_price) FROM fruits GROUP BY s\_id) GROUP BY s\_id;

**连接查询**

外连接内连接

内连接使用比较运算符将列的数据进行比较,然后筛选出符合条件的数据行

INNER JOIN 使用on作为条件判断

外连接

1.左连接LEFT[OUTER]JOIN 返回包括左表在内的所有记录和右表中与之连接字符相等的记录

2.右连接RIGHT[OUTER]JOIN返回包括右表在内的所有记录和左表中与之连接字符相等的记录

接字段相等的记录

位于连接条件左边的表,为左表,右边的为右表

**子查询**

在一个下查询语句内部嵌套查询语句

ANY 符合一个条件即可输出

ALL 符合所有条件才能输出

EXISTS 判断子查询的数据是否存在,存在执行外层查询,不存在返回空值,结束查询(判断子查询中内容是否为true,true执行外层查询,false不执行)SELECT怒骂

NOTEXISTS 判断子查询的内容是否不存在,不存在执行外层查询,存在返回空,结束查询

查看水果表中是否存在供应商的编号为107的数据,如果存在输出水果表中的高于12的水果记录

SELECT \* FROM fruits WHERE EXISTS (SELECT \* FROM suppliers WHERE s\_id = 107) AND f\_price>=12 ;

查询城市在北京的供应商都提供那些水果,显示供应商编号,水果名

SELECT \* FROM fruits WHERE fruits.s\_id=(SELECT s\_id FROM suppliers WHERE s\_city='Beijing');

COUNT(\*)和COUNT(字段)区别

count(\*)会计算表中数据最长列的数量,count(字段)如果该字段有空值将不会计入最后的结果

索引

索引是数据库中,对一列或者多列值进行排序的一种结构,用来提高数据库对特定数据的查询速度

索引单独存放在硬盘中｡索引基于存储引擎实现

索引的算法:

HASH:

BTREE:

Myisam支持btree索引

Memory和innodb支持btree和hash索引

每个种类的存储引擎的表至少支持创建16个索引,索引长度至少支持256个字节

索引特点

1 加快查询速度

2 实现数据完整性,加快表与表之间的联系 (两个表都要添加索引)

3 减少排序时间

4 创建唯一索引保证数据的唯一性

缺点

1 创建索引和维护索引需要消耗时间,随着数据的增多消耗量越大

2 索引占用磁盘空间

3 如果对于表中数据进行维护,删除,或者修改,索引也要动态维护

索引分类:

1 唯一索引和普通索引

普通索引:属于最基本的索引类型,存在的意义在于加快查询速度

唯一索引:在加快查询速度的同时,不允许在定义唯一索引的列上加入重复值

2 单列索引和组合索引

单列索引:基于单个字段添加的索引

组合索引:基于多个字段放到一起作为一个索引

3 全文索引

在定义全文索引的列上不允许写入空值

4 空间索引

同3

创建索引的规则:

1 索引不是创建的越多越好

2 数据量小的表最好不要创建索引

3 避免对经常跟新的数据创建索引

4 在不同值较多的列创建索引

5 如果唯一性是数据的特征时,创建唯一索引

6 在频繁进行排序的列上创建索引

在创建表的同时创建索引

CREATE TABLE 表名 字段名 数据类型 [unique 唯一索引 | fulltext全文索引 | spatial 空间索引] [index | key] [索引名] (定义索引的字段) [asc|desc]

分析查询语句

EXPLAIN 查询语句

普通索引

mysql> CREATE TABLE book

-> (

-> bookid INT NOT NULL,

-> bookname VARCHAR(255) NOT NULL,

-> authors VARCHAR(255) NOT NULL,

-> info VARCHAR(255) NULL,

-> comment VARCHAR(255) NULL,

-> year\_publication YEAR NOT NULL,

-> INDEX(year\_publication)

-> );

mysql> EXPLAIN SELECT \* FROM book WHERE year\_publication=1998\G

id: 1 查询语句标识符(查询的id 第一个查询1 第二个查询2)

select\_type: SIMPLE 查询类型,(主键查询,子查询,连接查询)

table: book 查询的表名

type: ref 连接类型 (表和表的连接关系)

possible\_keys: year\_publication 可能用到的索引

key: year\_publication 实际用到的索引

key\_len: 1 索引长度 (字节大小)

ref: const 表示字段或者常数和索引一起被使用(通常出现在用到索引的语句)

rows: 1 得到正确结果之前所要扫描的条数 (理想值1条)

Extra: Using index condition 额外信息

唯一索引(UNIQUE)

mysql> CREATE TABLE index1(id INT UNIQUE,

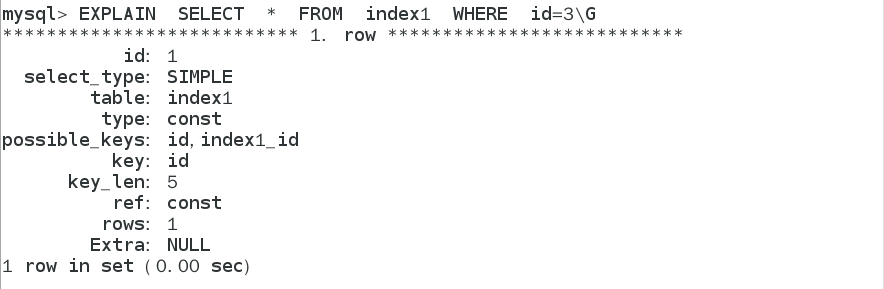
-> name VARCHAR(20),

-> UNIQUE INDEX index1\_id(id ASC)

-> );

mysql> INSERT INTO index1 VALUES (1,'a'),(2,'b'),(5,'c'),(4,'d'),(3,'e');

mysql> EXPLAIN SELECT \* FROM index1 WHERE id=3\G



单列索引

mysql> CREATE TABLE index2

-> (

-> id INT NOT NULL,

-> name CHAR(50) NULL,

-> INDEX index\_name(name)

-> );

组合索引

将多个字段放到一起建立一个索引,遵循最左前缀原则

最左前缀原则: 只有加上组合索引中最左侧的字段进行匹配才会用到索引

mysql> CREATE TABLE index3

-> (

-> id INT NOT NULL,

-> name CHAR(30) NOT NULL,

-> age INT NOT NULL,

-> info VARCHAR(255),

-> INDEX zhsy(id,name,age)

-> );

全文索引(fulltext)

只有MYISAM存储引擎支持,且只能给CAHR/VARCHAR/TEXT 列进行添加,在添加全文索引的列上,不允许写入空值,不支持局部索引

mysql> CREATE TABLE index4

(id INT NOT NULL,

name CHAR(30) NOT NULL,

age INT NOT NULL,

info VARCHAR(255),

FULLTEXT INDEX fullIdx(info(100)))ENGINE=MYISAM;

空间索引(SPATIAL)

添加该索引的字段不能为空值,且只有MYISAM存储引擎支持该索引

mysql> CREATE TABLE index5

-> (

-> g GEOMETRY NOT NULL,

-> SPATIAL INDEX index5\_g(g))ENGINE=MYISAM;

在已经创建好的表上添加索引

1 ALTER TABLE 表名 ADD [UNIQUE | fulltext | spatial] [index | key] [索引名] (定义的索引字段[length]) [asc | desc]

mysql> ALTER TABLE book ADD UNIQUE index uniIdx(bookid);

2 CREATE [UNIQUE|fulltext|spatial] index 索引名 ON 表名 (定义添加的索引字段[length]) [asc | desc]

mysql> CREATE INDEX book\_zu ON book (authors,info,comment);

删除索引

ALTER TABLE 表名 DROP index|key 索引名

使用DROP index

DROP INDEX 索引名 ON 表名

查看索引

SHOW INDEX FROM 表名\G

SHOW CREATE TABLE 表名\G

Seq\_in\_index: 该字段在索引中的位置(结合组合索引)

组合索引中如果将最左侧的字段删除的话索引依然生效,第二位的字段变为第一位继续提供索引的功能,依然符合最左侧前缀原则,直到所有的索引字段被删除,索引随着消失

存储过程

mysql> DELIMITER //

mysql> DROP PROCEDURE IF EXISTS test\_pro;

CREATE PROCEDURE test\_pro(num\_limit INT,rand\_limit INT)

BEGIN

DECLARE i INT DEFAULT 1;

DECLARE x INT DEFAULT 1;

DECLARE y INT DEFAULT 1;

DECLARE z INT DEFAULT 1;

WHILE i <= num\_limit do

SET x=FLOOR(RAND()\*rand\_limit);

SET y=FLOOR(RAND()\*rand\_limit);

SET z=FLOOR(RAND()\*rand\_limit);

INSERT INTO abc.test1 VALUES (NULL,x,y,z);

SET i= i+1;

END WHILE;

END//

联合查询

Select\_statement

Union[all]

Union select\_statement

将联合查询(union)前后两个查询语句的结果进行结合,最终显示

select \* from fruits union all select \* from fruits;

All是否过滤重复数据,如果写all不过滤重复数据,如果不写则过滤

select f\_name from fruits union all select f\_price from fruits;

联合查询的前后两个查询语句没有任何联系,不会区分数据类型,数据类型不一致同样可以输出

在联合查询的过程中字段的数量需要一致,否则报错｡

**视图**

一张虚拟表,和真实表一样,试图中包含带有名称的行和列的数据,试图可以从一个表或者多个表获取试图从5.0版本之后应用,用于保障数据安全,视图一经创建就会存放到数据库中

视图的优点

1.简单化:看到的就是需要的

2.安全性:用户只能通过视图看到想让用户看到的

3.逻辑结果独立性:视图可以屏蔽真实表结构变化带来的影响

创建视图的命令

CREATE [OR REPLACE] [ALGORITHM视图的选择算法={UNDEFINED|MERGE|TEMPTABLE}]VIEW 视图名[(定义视图的字段)] AS sleect\_statment[with[cascaded|local]check option]

[OR REPLACE]替换

ALGORITHM 视图算法

UNDEFINED 视图将会自动选择算法

MERGE 合并算法

TEMPTABLE 先将视图的结果存放到临时表然后根据临时表执行语句,临时表算法

cascaded 更新视图时需要满足所有视图和所有的表的相关条件

local 更新视图时许哟啊满足自身条件即可

单表视图:视图中所有的数据从一个表中导出

多表视图:视图中的数据从多个表中导出

1.视图可以通过ddl和dml语句进行操作,(insert,update,delect,alter)

2.视图中不存放数据

3.视图中的数据被改变真实表数据也会随之发生改变,反之亦然

4.视图可以基于视图进行建立

5.视图中不包含存储引擎以为这不能添加索引

5.7版主之前如果使用cascaded语句,只会向上层进行筛选,基于该视图之后建立的视图cascaded不会进行判断,5.7版本之后,上下层所有的视图以及真实表的条件都讲列入判断行列

查看视图信息

SHOW TABLE STATUS LIKE ‘视图名’ \G

Engine: NULL 存储引擎

Version: NULL 版本

Row\_format: NULL 行格式

Rows: NULL 表中条数

Avg\_row\_length: NULL 平均每行包含的字节数

Data\_length: NULL 整个表的字节数

Max\_data\_length: NULL 表中可以村放的最大字节数

Index\_length: NULL 索引长度

Data\_free: NULL 剩余空间

Auto\_increment: NULL 下一个将要自增的值

Create\_time: NULL 创建该表的时间

Update\_time: NULL 最后一次修改的时间

Check\_time: NULL 最后一次检查表的时间

Collation: NULL 排序规则

Checksum: NULL 保存表的实时校验项

Create\_options: NULL 额外信息

.frm为表的结构文件

SHOW CREATE VIEW 视图名\G

修改视图

CREATE OR REPLACE [ALGORITHM视图的选择算法={UNDEFINED|MERGE|TEMPTABLE}]VIEW 视图名[(定义视图的字段)] AS sleect\_statment[with[cascaded|local]check option]

ALTER [ALGORITHM视图的选择算法={UNDEFINED|MERGE|TEMPTABLE}]VIEW 视图名[(定义视图的字段)] AS sleect\_statment[with[cascaded|local]check option]

with check option=with cascaded check option

如果不写[with[cascaded|local]check option]=local 满足自身条件

视图中包含一下情况的字段无法被更新

1.视图中不包含真实鸟中被定义为非空的列

2.定义视图的select语句后使用数字表达式

3.定义视图的select语句后使用聚合函数

4.定义视图的select语句后使用union/having/group by 无法更新

如果真实表中删除了放入视图中的字段，视图将会随之消失，除非修改视图或者真实表重新添加同名字段,或者删除视图,否则视图将一直不能使用,且会影响数据的备份

删除视图

drop view [IF EXISTS]视图名

视图和真实表有什么区别

1.视图属于编译好的sql语句，是数据集成的可视化的表，而表不是

2.真实表存在数据，占用空间，而视图属于虚表，不存放数据，占用极少的空间

3.视图的建立或者删除直影响视图本身，不影响基本表

4.视图属于查看表的方式，可以从一个或者多个表中导出，用于保证数安全的，而真实表是由表结构和数据的

5.视图没有存储引擎，不能建立索引，而真实表拥有存储引擎可以建立索引

视图和真实表的联系

视图基于真实表存在的，虽然视图可以基于视图进行创建，但是最终开始的

视图也是基于真实表建立的，如果没有真实表，不能创建视图

.MYD 使用MyISAM存储引擎是出现，用于存放数据

.MYI 使用MyISAM存储引擎出现，用于存放索引信息

**事务**

事务是一个多条sql语句的集合要么都成功，要么都失败

事务的实现方式通过存储引擎

Mysql事务含有acid四大特性

A原子性（atomicity）一个事务必须视为一个不可分割的单元

C一致性（consistency）数据库从一种状态切换到另一种状态

I隔离性（isolation）事务在提交之前对其它事务不可见

D持久性（durablity）一旦事务提交，所修改的数据永久的保存到硬盘

事务中的命令

开启事务bgngin 或者start transaction

回滚 rollback 回到事务最开始的状态

提交事务commit

当输入两次begin时，如果autocommit为开启状态，mysql会将上一次的事务进行提交，然后开启新的事务。

set autocommit=0；

事务保存点

savepoint名字

回滚

rollback to 保存点名

隔离级别

1.未提交读read-uncommitted

问题

脏读:事务在未提交时,对其他事务可见

幻读:查看的数据和最后的结果不一致

不可重复读:不能将之前的数据重复读取

2.已提交读read-committed

依然存在幻读问题

解决了脏读的问题

不可重读读依然存在

3.可重复读(生产环境中常用的隔离级别

解决了不可重复读和脏读的问题

幻读依然没有得到解决

4.可串行读

作为等级最高的隔离级别,可串行读解决了上述的问题

可串行读将事务进行排序,逐个执行

缺点:速度非常慢,效力低

锁

读锁:如果一个事务对表A加了读锁,那么其他事务可以继续对A加读锁,但是不能添加写锁,知道释放了对A的加锁

写锁(排它锁,x锁)如果一个事务对biaoA加锁,那么其他事务不可以加任何锁,直到释放对A的加锁

表锁:属于并发最低锁策略,相当于如果事务对A加锁,其他事务可读不可写,一旦查看了表的所有的内容,整个表锁住不允许用户修改

行级锁:当用户使用数据时,只锁住用户修改的数据,其他用于依然可以访问没有被加锁的数据,但是开销较大,可以处理高并发

SELECT @@global.tx\_isolation,@@tx\_isolation; 查看数据默认设置隔离级别

@@tx\_isolation 当前数据库设置的隔离级别

触发器

trigger属于一个特殊的存储过程,属于嵌入到mysql的一段程序,且触发器是由时间进行触发(出触发时间,:DML语句 insert\update\delect),一旦通过触发时间将触发,将会执行预先再触发器中这只好的内容,整个过程自动执行,不需要手动触发

创建触发器的语法

create trugger 触发器名 触发时机(before或者after)触发事件(insert\update\delete) ON 表名 FOR EACH ROW [END 在有复杂触发过程的触发器中的使用]

UPDATE 表名 SET 字段名=要修改的内容(可以是数据,可以使用函数) [WHERE 条件判断]

触发器

CREATE TRIGGER test\_tri1 BEFORE INSERT ON test1

FOR EACH ROW BEGIN

INSERT INTO test2 SET a2=NEW.a1;

DELETE FROM test3 WHERE a3=NEW.a1;

UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;

END//

一个表中只能创建两个触发器

分别是触发时机为before和after各一个

before设after用于表示触发事件,before表示触发器中的命令dml修改语句之前执行

after表示触发器中的命令再dml修改语句之后执行

查看触发器的方式

show triggers

或者 SELECT \* FROM information\_schema.TRIGGERS\G

删除触发器

DROP TRIGGER 触发器名

触发器是基于数据表创建的如果想要删除触发器,需要在对应库中进行删除

或者使用指定库名的方式进行删除

存储过程和触发器的区别

触发器始于特殊的存储过程,不需要手动触发,当符合触发事件的条件自动隐士触发

容易在不知情的情况下更改其他数据,而存储过程需要使用CALL命令进行触发,属于指定触发方式,避免通过触发器误改数据

**用户权限**

user表位于mysql库中,

如果想要查看用户权限

SELECT \* FROM mysql.user 直接查看用户信息以及用户的权限

用户列

host 主机,user 用户名,password密码三部分,当链接数据库时,只有匹配了上述三个字段的所有信息才能连接数据库,否则登录失败

user host 作为联合主键负责确定位于的用户信息

两个ssl关于加密选项,两个x509挂怒识别用户身份,plugin识别用户身份的插件,authentication\_sting(5.7版本之后用于替代password字段用户显示密码)

权限列

表中以priv结尾的内容叫做权限列,权限列决定了用户是否权力执行对应的操作(产看权限,写入权限)如果有将可以执行对应权限,没有则不能,类型为enum枚举,结果分为Y(yes)和N(no)

资源控制列

资源控制列意味着将用户可以使用的资源进行限制,防止单个用户过多使用服务器资源

max\_questions:0 用户每小时被允许执行查询操作的次数

max\_updates:0 用户每小时被允许执行更新操作的次数

max\_connections:0 用户每小时被允许执行连接操作的次数

max\_user\_connections:0 用户被允许同时建立连接的次数

flush privileges； 刷新权限，让mysql重新识别user表中数据（一般情况下只有直接操作user表需要使用该条命令，其他创建用户的方式不需要使用该命令）

Select\_priv: Y 查询权限

Insert\_priv: Y 写入权限

Update\_priv: Y 更新权限

Delete\_priv: Y 删除权限

Create\_priv: Y 创建权限

Drop\_priv: Y 删除权限

Reload\_priv: Y 是否可以使用flush权限

Shutdown\_priv: Y关闭mysql进程的权限

Process\_priv: Y 显示和中断的权限

File\_priv: Y 载入文件的权限

Grant\_priv: Y 给其他人授权的权限

References\_priv: Y 对于约束进行操作的权限

Index\_priv: Y 索引权限

Alter\_priv: Y 修改权限

Show\_db\_priv: Y 查看数据库权限

Super\_priv: Y 超级权限

Create\_tmp\_table\_priv: Y 创建临时表权限

Lock\_tables\_priv: Y 锁表权限

Execute\_priv: Y 执行函数和存储过程的权限

Repl\_slave\_priv: Y 对于复制(slave)操作的权限

Repl\_client\_priv: Y 查看master和slave权限

Create\_view\_priv: Y 创建视图权限

Show\_view\_priv: Y 查看视图权限

Create\_routine\_priv: Y 创建函数和存储过程权限

Alter\_routine\_priv: Y 修改函数和存储过程权限

Create\_user\_priv: Y 创建用户权限

Event\_priv: Y 操作事件权限

Trigger\_priv: Y 对于触发器进行操作的权限

Create\_tablespace\_priv: Y 创建临时表权限

创建用户

1 create user ‘user’@’host’ [identifled by [password] ‘password’]

user 创建用户登录的身份

identified by 创建密码

如果加password 以hash方式设置密码

2.grant privileges on db.table to user@host [identified by ‘password’] [,user2@host[identified by ‘password by’]][with grant option]

privileges 赋予用户指定权限

db.table 表示该权限在那个库哪个表上生效

identified by 设置密码

with grant option

一旦用户通过grant命令授予权限是不能通过grant收回需要用特殊的revoke命令收回权限.

授予权限一般情况下时成组存在的,如果仅仅赋予一个权限,可能会造成该用户无法完整指定操作的问题

crrate user和grant 区别

garant 创建用户,如果该用户存在依然可以执行语句相当于赋予权限,如果不存在直接创建用户,create user不能赋予权限

3.create user 不能修改用户密码,而grant可以修改用户密码

4.create user一次性只能创建一个用户,grant 可以同时创建多个用户,并且赋予权限

with grant option 如果在创建时,向赋予创建的所有用户给其他人授权的权限,那么只需要在最后写一条with grant option 就可以

直接擦欧总user表

删除用户

1.DROP USER 用户名@主机名

2.直接操作user表

修改密码

1.mysqladmin

mysqladmin -u user -p 旧密码 password 新密码

清空密码:mysqladmin -u user -p 旧密码 password ‘’(执行改操作需要有super权限)

2.直接操作user表

3.set password(直接修改自身密码,修改其他用户的密码)

改自己set password=password(‘密码’)

改别人 set password for ‘user’@’host’=password(‘密码’)

4.使用grant 修改密码

grant privileges ON db.table TO ‘user’@’hsst’ identifiled by ‘passwird’如果用户不存在直接创建用户赋予权限,设置密码,如果用户存在,可以更改权限,如果权限不变可以修改密码

如果密码忘记,找回密码

1.受用mysqld\_safe –skip-grant-tables –user=mysql 使用跳过授权表示步骤

2./etc/init.d/mysqld restart –skip-grant-tables 重启mysql

3.

4.

5.

收回权限

1.收回所有权限

REVOKE ALL PRIVILEGES,GRANT option FRM ‘user’@’host’[‘user’@’host’]

2.收回指定权限

REVOKE priv,priv…ON db.table FROM ‘user’@’host’

查看用户权限

1.直接查看user表

2.SHOW GRANTS FOR ‘user1’@’HOST’\G

usage 表示用户可以通过该权限连接数据库,随着用户的创建产生直到用户被删除位置,且不能通过revoke 进行收回在user表中也查看不到,只能通过show grant 方式进行查看

user查看表示用户的权限,至于某个单位表的权限不会体现在user表中

通过show grant for 可以查看部分表上的权限

**mysql日志**

mysql记录了musql数据库的日常操作和错误日志,不同的日志记录的不同的内容,在mysql一共有四大类日志

1.错误日志:记录了mysql运行时,启动和停止时的错误

2.查询日志记录建立的客户端连接执行的语句

3.二进制日志记录所有数据库进行修改的语句,用于灾难恢复和复制

4.慢查询日志记录所有执行事件超过阈值的查询语句

log\_bin | ON 是否开启二进制日志

log\_bin\_trust\_function\_creators | OFF 是否允许将自定义函数或者存写入二进制日志

log\_error | /usr/local/mysql/data/localhost.localdomain.er 错误日志的保存位置

log\_output | FILE 日志的输出形式

log\_queries\_not\_using\_indexes | OFF 是否记录没有使用索引的查询语句

log\_slave\_updates | OFF 是否允许从库更新

log\_slow\_queries | OFF 是否开启慢查询日志

log\_warnings | 1 是否将警告记录到错误日志

除了二进制日志是一个二进制日志文件其余三个日志全部为文本文件可以使用vim进行编辑

刷新日志

1. mysqladmin -u user-p 密码 flush-logs

2.在mysql 命令行使用flush logs

3.重启mysql服务

删除错误日志可以通过flush logs进行找回

二进制日志

默认情况不开启,包含了对数据库的更新信息(ddl喝DML语句),不包含没有对数据库产生变更的语句,使用二进制日志最大的目录是复制喝尽可能的恢复数据库(灾难恢复),且事务安全的方式进行记录数据

xpire\_logs\_days=5 设置二进制自动清除天数

max\_binlog\_size=300M 每个日志的大小,如果不设置默认1G

SHOW BINARY LOGS; 查看二进制日志

删除二进制日志的命令

reset mater

purge master logs to ‘二级制日志’

purge master(binary) logs before ‘date’

查看二级制日志内容

mysqlbinlog二进制日志名称

恢复数据

mysqlbinlog [option]二进制日志名|mysql -u user -p 密码

option 中选项

--sart-position 开始的偏移量

--stop-position 结束的偏移量

--start-date 开始的日期

--stop-date 结束的日期

指定结束的偏移量(结束的日期)表示该二进制日志文件开头的偏移量(日期)进行恢复数据

通过日期恢复数据格式要求

填写分隔符]

yyyy-mm-dd hh:mm:ss

不写任何分隔符

yyyymmddhhmmss

id name age 3个字段 id为之间

数据

user 20

user2 25

user30

删除整张表然后恢复数据,(根据偏移量事件各恢复一次

复制过程中master\_log\_file 和master\_log\_pos中一定写入主的二进制日志和偏移量,不能写老二进制的名字,否则辅助出问题

show master start 看到最新的主的偏移量以及使用二进制日志文件

查询日志:记录客户端用户所作的操作

general\_log=ON 5.6 版本开启查询日志的方式

log 5.5版本开启查询日志的方式

**优化**

解决mysql瓶颈,提高mysql的速度,让系统提高更大负荷的服务优化查询语句

EXPLAN 分析查询语句,找出mysql的瓶颈,帮助运维人员或者dba进行优化

Select\_type:simple不包含union或者子查询

Primary 表示该查询为最外层的查询，或者主键查询

Union 表示为连接查询的第二个查询语句或者之后的查询语句

Union result 联合查询的结果

Subquery 子查询中的第一个select

Type：判断索引是否高效，通过type字段可以确定是全表扫描还是索引扫描

System:系统表，表中一般情况下只有一条数据，属于特殊的const类型

Const ：针对主键或者唯一索引使用的等值查询扫描，只返回一条数据，速度非常快

id:1 select语句标识符

select\_type:SIMPLE 查询类型

table:user\_info 连接类型

type:connst 可能用到的索引

possible\_key:PRIMARY 实际用到的索引

key\_len:8 索引长度

ref:const 字段或者常量和索引一起被使用

rows:1 得到正确结果之前索要扫描的条目数

Extra: 额外信息

select\_ttype:常见类型

simple:简单查询,不包含union查询或者子查询

primary:表示此查询时最外层的查询或者主键查询

union:次查询是联合查询的第二个或者之后的查询

union result:联合查询的结果

subquery:子查询的第一个select

type

该字段作为判断查询是否高效的依据,可以查看索引是否高效

system 表中只有一条数据,属于特殊的const类型(系统表)

const 针对主键或者唯一索引的等值查询扫描,最多返回一条数据

eq\_ref 通常出现在多表连接查询,表示对于前表的一个结果,都能匹配到后表的一个结果,且比较运算符是使用=

ref 通常出现在多表连接查询,针对于非主键或者非唯一索引,或者使用了组合索引的查询

range 范围扫描,一般出现使用比较运算符的查询语句中

index 索引范围扫描,出现在查询的内容在索引树直接就能获取,不需要扫描数据,但是需要扫描全部索引

all 全表扫描,属于效最低的扫描方式,对于你运维或者dba来说是不允许出现all了类型,如果出现,没有索引需要添加索引或者修改查询语句

性能

ALL<index<range<eq\_ref<const<system

key\_len表示索引长度,可以通过该字段确定组合索引是否完整生效,或者只有部分生效

索引长度计算规则(utf-8)

varchar(n) 占用3n+2字节 date 3字节

char(n) 占用n字节 time 3字节

TINYINT 1字节 timestame 4字节

SMALLINT 2字节 datatime 8字节  
MEDIUMINT 3字节

INT 4字节

BIGINT 8字节

YEAR 1字节

如果字段属性含有可以为空的内容,需要占用1字节的空间,如果为not null 则不会占用空间

ken\_len在使用比较运算符时(<>)被截断,只会得到最左侧的字段的所占用的字节数,速度相应快一些,如果使用=匹配可以使用所用的包含在组合索引中的字段进行匹配,得到数据相对精确

extra 可以出现的信息

using filesort mysql需要额外排序的

using where 使用where条件判断

using index 使用索引

生产环境中避免使用子查询，尽可能将子查询用连接查询替代

使用模糊查询,%不要放在开头,将%放在后面否则不会使用索引

当使用OR条件时,只有or前后所有得判断条件都有索引最后才会使用索引,索引不生效

**优化表结构**

1.将字段很多得表分成多个表

2.添加中间表

一次性将复合要求得数据写入到中间表

3.合理添加冗余字段

优点,增加冗余字段可以让数据变得容易查找

缺点:1占用双倍空间

2.数据有可能不一致

优化写入数据得速度

1.禁用索引 alter table 表名 disable keys enable=开启

对myisam存储引擎

2.禁用唯一得检查 set unique\_check=0 关闭 =1开启

3.禁用外键 set foregin\_key\_checks=0 关闭 =1开启

4.禁用自动提交 set autocommit=0 关闭 =1开启

5.使用批量插入数据方式

独享表空间ＩＮｎｏｄｂ　的表数据放在ｉｄｄａｔａ１中，如果数据量将会造成文件过大，２０Ｇ造成读取数据缓慢，独享空间的作用，解决将数据放在一个文件中造成文件过大问题，每个表变更ｉｂｄ文件用于存放表的数据，不会再写入ｉｂｄａｔａ１中

备份

避免因为误操作或者系统突然宕机造成的数据丢失问题；需要将数据进行复制操作，将该复制出来的内容放入其他的存储设备中

备份方式：

1）：mysqldump -u user -ppasword 【option】db 【表1，表2，表3....】> /path

-u：指定用户

-p：指定密码

-R：只导出存储过程和函数

-d：只备份表结构不备份表数据

-t：只备份表数据不备份表结构

-A：备份所有库

-B：备份多个库

--opt：一次性添加--add-drop-table（在创建表之前增加drop table if exists判断语句，防止表重名）；--add-locks（在insert之前和insert之后进行锁表）；--extended-insert（可以使用多行插入）；--lock-table（在备份时锁表）

2）直接复制数据--cp

3）使用mysqlhotcopy：perl语言写的备份工具，只能备份表。

mysqlhotcopy -u user -p password 【-S sock-pth】 db < /path

sock-pth：/tmp/mysql.sock

还原：1）mysql -u user -ppassword 要还原的库名 < /path

2）直接mysql命令行写入：source /path

3）如果通过cp或者mysqlhotcopy进行备份数据，直接将备份出来的目录或者文件移动到指定的数据目录下，重启musql即可恢复数据

生产环境中，常用mysqldump进行全量备份；使用xtrabackup进行增量备份。

create or replace view 视图名 as select \* from 表名：修改视图

**redis**

redis（remote dictionary server）：基于键值对的数据结构服务器，属于nosql（非关系型数据库）

数据类型：

字符串（string）

hash

列表（list）

集合（set）

有序集合（zset）

特点：redis工作在内存中，由于内存的特质，读写速度将会非常快，达到10万次/s

1）快---工作在内存中，效率高；由c语言编写，距离操作系统近，执行速度快；属于单线程架构，没有其他线程争抢资源

2）基于键值对的数据结构服务器（字典结构）

3）功能丰富

4）简单稳定

5）客户端语言多

6）持久化（aof独立日志和rdb快照）

7）复制

8）高可用

bitmaps（位图），hyperloglog（流量统计），geo（地理信息定位），事务，流水线，持久化，键过期，复制，哨兵

redis-sserver：服务端启动程序

redis-cli：客户端启动程序

-h：指定主机

-p：指定端口

redis目录下src目录下：

redis-check-aof：aof持久化的检查修复工具

redis-check-rdb：rdb持久化的检查修复工具

redis-sentinel：哨兵的启动工具

redis-benchmark：redis基准测试工具

redis.comf配置文件：

port 6379 ：默认端口，可更改

bind 127.0.0.1：允许连接redis的主句；0.0.0.0：表示任何人可连接

daemonize no：是否后台运行，yes/no

logfile ""：指定日志文件的路径

如果开启后台daemonize yes 如果需要保留日志将logfile选项打开

如果关闭后台daemonize no 需要将logfile的内容清除否则输出不了日志

关闭redis进程：

redis-cli -h 127.0.0.1 -p 6379 shutdown    客户端程序默认连接127.0.0.1下的6379端口，如果配置文件端口被修改，需要加-h，-p选项

直接在redis客户单执行shutdown

全局命令：

keys \*：查看所有键值对

set key value：设置键值对（查看键值对内容：get key）直接复制数据

判断键值对是否存在,存在返回1,不存在返回0

exists key

删除键值对(同时可以删除多个)

del key key...

redis没有原子性,键值对的存放是无顺序的

dbsize

设置键值过期

expire key seconds 给指定的键值对设置过期时间,一旦达到过期时间redis将自动的把键值对进行删除

查看过期时间

ttl key

结果>0 剩余过期时间

结果-1 键值对存在没有设置时间

结果-2 该键已过期或者不存在该键值对

如果对于一个已经设置完成过期时间的键值对重新设置过期时间,之前的剩余过期时间将会被覆盖(键没有删除)

查看键值对的数据类型

type key

内部编码,每个数据类型对应一个或者多个内部编码,每种数据类型底层内部编码实现

string 字符串 int embstr raw

hash 哈希 hashtable ziplist

list 列表 quicklist intset

set 集合 hashtable intset

zsset 有序集合skiplist ziplist

内部编码优点

1.改进内部编码对于数据类型和命令没有影响

2.不同的内部编码可以在不同常景下发挥各自作用

查看内部编码的命令

object encoding key

字符串数据类型

属于redis最基本的数据类型

其余集中数据类型都是基于string的基础之上进行开发的

字符串中的数据可以是文本,也可以是二进制数据(图片,视频,音频)且每个字符串的最大存放空间为512字节

set key value [ex seconds][px milliseconds][nx|xx]

ex 设置秒级别的过期时间

px 设置毫秒级别的过期时间

nx 健必须不存在才能设置成功

xx 健必须存在才能设置成功

如果对于一个已经存在的键值对继续使用set命令将会把之前的内容覆盖

设置字符串数据类型

set key value

查看字符串数据类型的值

get key

批量设置键值对

mset key value key2 value2 …

批量查看键值对

mget key key2 key3

计数

incrbyfliat key 小数 对于键值对中的数据进行增加(可以增加浮点数)

追加值

append key value 将值在不覆盖源数据的情况下追加

incr key 对于键值对中数值进行增加(键值对中必须为整数且只能有整数)

dect key 对于键值中的数据进行减少()

incrby 数值 同incr

decrby 数值 同decr

检测字符串的长度

strlen key

登录时使用-raw 将汉字正常显示(一个汉字占用3字节空间)

设置并返回原值

getset key value

设置指定位置的字符

setrange key opsition (位置) value

如果写入的位置超过字符串的最大长度时,redis会再数据和更改为值得字符串之间填写\x00(零字节)只占用空间,没有实际意义

int 长度为8字节得长度整形

embstr 不超过44字节得字符串

raw 超过44字节的字符串

关闭redis服务器

1.redis-cil -h host -p port shutdown

2.在redis命令行输入shutdown

开启多个redis实例

在mysql中通过编辑配置文件添加[mysqld]来实现通过不同的端口开启多个mysql

在redis中通过复制配置文件,修改port端口来进行开启多个redis实例(不建议将redis配置文件复制到同一个目录)

哈希

key field value redis的哈希采用字典结构，存放了键于值的之间饿映射，包含了字段和值之间的映射，在hash的类型中一个键值对最多可以存放2的32次方-1子字段

键 字段 值

设置hash的键值对

hset key field value

删除字段 field

计算字段的个数

hlen key

批量设置键值对

hmset key filed value key field value……

查看所有的field

hkeys key

查看所有的value

havis key

查看所有的字段和值

hgetall key（如果字段过多会造成阻塞）

给hash字段中的数据进行数量增加（仅限数值）

hincrby key field 数值

hincrbyfloat key field 数值

计算字段中值的长度

hstrlen key field

hash 内部编码

ziplist压缩列表，当哈希的元素个数小于512个或者所有字段的值都小于64字节，使用ziplist作为内部编码，可以更加进错的压缩数据，实现数据的连续存放hashtable哈希表，当不满足ziplist条件时，使用哈希表作为内部编码

list列表

可以存放有多个有序字符串，每个list可以存放2的32次方-1个字符串（元素）

1.元素是有序的（元素下标）

2.元素可以重复

操作 命令

添加rpush lpush linsert

查看lrange lindex llen

删除lpop rpop lerm ltrim

修改lsetke befor after pivonetvalue

建立列表键值对

rpush key value value... 右写入

lpush key value value... 左写入

linser

lisert keyfore dfater pivot netw irj’

查看

irange jstart恶end

删除lgop rop lrem ltrim

lgop key 从左边删除，每次删除以恶搞（左弹出）

rgop key 从右边删除，每次删除一个（右弹出）

删除指定元素

lrem key count value

coount>0 从左往右删，删除count个value

coount>0 从右往左删，删除count个value

count=0 删除所有value

范围删除元素

ltrumkey start stop 删除指定元素范围我意外的所有元素

修改

lset key index newvalue 索引下标不能超过当前元素的索引下标否则报错

集合（set）

集合用力存放多个字符串元组，但是和列表不相同，集合中不允许出现重复的元素，一个集合中最多可以存放2的32次方-1个晕元素，且结合中可以求交集，并集、差集

1）集合中的元素没有顺序

2）集合中的元素不允许重复

内部编码

列表中只有一个内部编码

quicklist 属于linkedllst和ziplist的集合，一个双向链表结构，在实现数据的有序存储上，实现紧凑的存储模式，达到快速查找的目标

添加集合

sadd key element element…

删除元素

srem key element element…

计算元素个数

scard key

判断元素是否在集合中

sismenmber key element

从集合中随机删除指定的个数元素

spspkey [count]

查看 集合中中搜所有的元素

smembers key

集合之间的操作

交集 sinter[store]sinter key key…

并集 sunion[store]sunion key key…

差集 sdiffp[store]sdiff key key …

destination 保存数据的集合名

集合内部编码

intset：当集合中的元素都是帧数且个数小于512时，使用intser作为内部编码

hashtable：当集合中的元素不满足intest条件，使用hashtable作为内部编码

有序集合

对于有集合而言，依然保留者某种集合中的特点，集合中不允许出现重复的元素，但是元素实现有序化，和列表又不同，有序集合实现数据有序的方式不是索引下标，而是score（分值）

zset在set的某基础之上给每一个元素增加一个分值，一个元素对应一个分值，且分值必须时数值

键名，分值，元素

建立有序集合

zadd key score member [score member]

zadd key 【nx】【xx】【ch】【incr】score member 【score member】：建立有序集合

nx：不存在才能设置成功

xx：存在才能设置成功

ch：操作之后有序集合重元素和分数发生变化的个数

incr：对指定的元素的分值进行增加

zcard key：计算成员的个数

zscore key member：查看某个成员的分值

zrank key member：由低到高查看成员排名

zrevrank key member：有高到低查看成员排名

删除元素

zrem key member1 member2…

增加指定成员的分值

zincrby key imcrement menber

查看指定排名范围的成员

zrange key start end [withscores] 由低到高查看

zrevrabge key start end [withscores] 由高到查看

查看指定值之范围内的成员

zrangebyscore key min max [withscores][timit offset count] 由低到高查看

zrevrangebyscore key max min [withscores][timit offset count]由高到低查看

withscore 查看时带分值显示

limit offset count 限制输出的条数

+inf 正无穷 -inf 负无穷

查看指定分数范围内的成员个数

zocunt key min max

删除指定排名内的元素

zremrangebyrank key start end

删除指定分数范围内的元素

zremrangebyscore key min max

有序集合之间的操作

交集

zintrestore destination numkeys key key... [weights weight][aggregate sum|min|max]

并集

zunionstore destination numkeys key key... [weights weight][aggregate sum|min|max]

destination 用于保存结果的键值对的名字

numkeys 做交集或者并集的集合数量

key 求交集的集名

weights 设置每个健的权重

aggregate 求完交集或者并集之后分值按照最大值|最小值|总和进行选择

内部编码

skiplist(跳跃表)当不满足ziplist条件,使用skiplist实现数据的更优存储

ziplist(压缩表)当有序集合的元素个数小于128哥且每个元素小于64字节

使用ziplist作为内部编码,实现数据的紧凑压缩

键的重命名

rename key newkey

如果将键值对改名时,新的名字如果已经存在当前数据库,将会把之前的内容全部覆盖

查看当前键值对的个数

dbsize

键过期

expireat key 时间戳

数据库管理

1.切换数据库

select dbindex

2.flushall/flushdb

清空所有数据库(0-15)all

清空当前所有的数据库 db

--rdb 将现在数据库的内容拍摄成快照保存在硬盘中(持久化方式的一种)

RDB(快照) AOF(日志)

持久化可以有效的避免因为进程的退出或者不可抗力造成的数据丢失问题(突然断电)当重启redis时,每次可以通过保存在硬盘中的持久化文件来恢复数据

ROB将数据生成快照保存在硬盘中,且触发的方式分为手动和自动

手动触发

redis 中命令 save | bgsave

通过cli 客户端实现rdb持久化

redis 阻塞当前服务器,知道持久化完成结果阻塞

bgsave redis 进程通过fork 操作创建子进程,reb 持久化由子进程负责,且子进程完成后自动退出

自动触发

1.在m秒内修改n次键值对会触发ggsave

2.如果从节点执行全量复制主节点自动bgsave

3.如果执行debug reload 触发save操作

4.执行shutdown,如果aof持久化没有开启自动bgsave

通过快照恢复数据时,在众多中,系统恢复配置文件中dbfilename后名字一样的文件数据,可以在一个目录下保存多个快照,名字buweidbfilename设定为名字即可(默认dump.rdb)

bgsave工作流程

1.当输入bgsave时,父进程会进行判断当前是否执行持久化的子进程,如果有直接返回,不做任何操作,如果没有,则进行计划

2.父进程通过fork操作生成子进程,子进程负责持久化

3.父进程弯沉fork操作之后,可以继续响应其他命令

4.子进程生成rdb文件,根据父进程内存生成快照文件,替代原有rdb文件

5.子进程告知父进程任务完成,自动退出

Rdb优点

Rdb属于紧凑压制得二进制文件，表示在某个时间节点得数据快照，适用于全量备份，全量复制

二进制读取速度要高于aof文件（日志）

缺点

1.rdb没有办法做到秒级别备份,因为拍摄快照(生成子进程属于重量级操作)

2.兼容性问题

AOF持久化

以独立日志的形式保存每次写入的命令,重新启动时再一次执行,aof文件中的命令到恢复数据的母的,解决了秒级别备份问题,属于主流性值的持久化方式,默认不开启

aof工作流程

1.命令写入append

2.文件同步sync

3.文件重写rewrite

4.重启加载load

1.所有的命令写入aof buf 缓冲区

2.aof文件过大,通过重写加载缩小体积

3.redis重启,可以加载aof文件恢复数据

aof使用文本格式

1.兼容性优越于二进制文件

2.文本协议具有可读性可以直接修改和删除

为什么命令追加到aof缓冲区

1.redis属于单线程响应命令直接追加到硬盘,性能将会取决于硬盘效率变低,如果写入缓冲区,缓冲区可以通过策略向硬盘进行同步

同步策略

always:命令写入就同步到硬盘

everysec命令写入缓冲区调用write操作每秒调用一次(达到秒级别备份的关键因素)

no命令写入后就不做同步操作,同步由操作系统完成

由于命令的写入,文件越来越大,包含有用或者无用的数据,redis使用重写机制缩减aof文件的大小

为什么aof文件通过重写可以缩减体积

1.超时的数据不会呗记录

2.旧的命令不再被执行

3.多条命令合并为一条

重写触发机制分为手动和自动两种

手动bgrewriteaof

自动

根据配置文件进行触发

auto-aof-rewrite-percentage 100 当前的aof文件的大小和上次重写aof文件大小的比值

auto-aof-rewrite-min-size 64mb 重写过后aof文件的最小体积

子进程消耗

内存

子进程通过fork操作生成，理论上占用和父进程一样的内存，但是由于写时复制（copy-on-write）技术的诞生，可以共享父进程物理内存页，子进程只会消耗修改数据部分的内存

硬盘

生成持久化文件要向硬盘大量写入数据，对于硬盘来说数据密集操作，尽可能不要和消息列队服务器和存储服务（大型数据）部署在一起

Cpu

对于单核cpu来说 子进程将会消耗单核cpu的90%以上，不要做redis绑定单核cpu操作

**OneProxy**

1.垂直切库

2.水平切表

3.proxy群集

4.读写可用

5.读写分离（master不参与读写）

6.读写分离（master参与读取）

7.写高可用

8.读写随机

垂直切库 降低读写压力

水平切表 降低写入压力

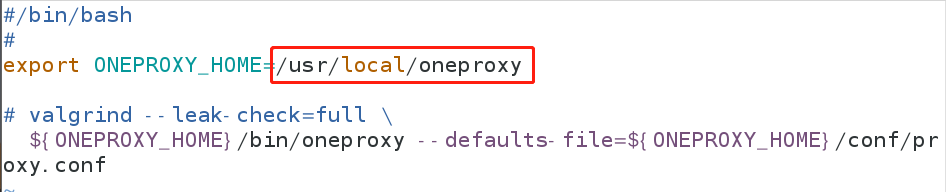
Oracle 的安装部署

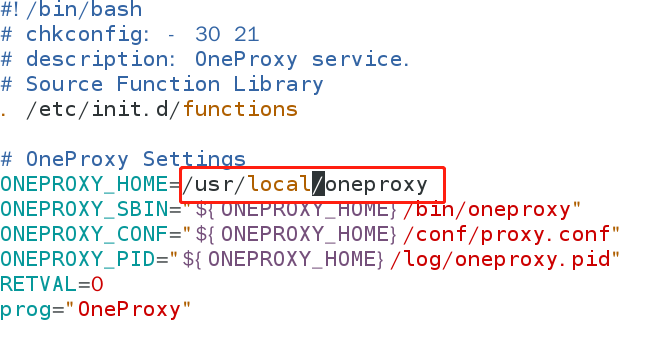
解压oneproxy

tar -zxvf oneproxy-rhel5-linux64-v6.0.0-ga.tar.gz -C /usr/local/

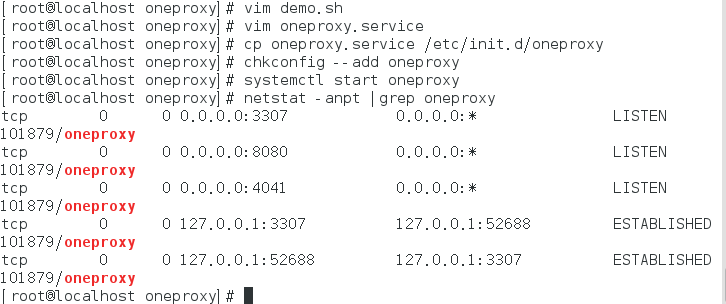
编辑配置文件修改路径





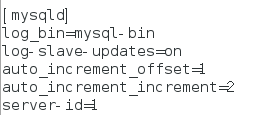


复制oneproxy到etc/init.d/oneproxy



让第二台服务器成为主节点服务器三成为从节点

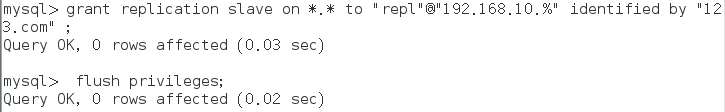
主

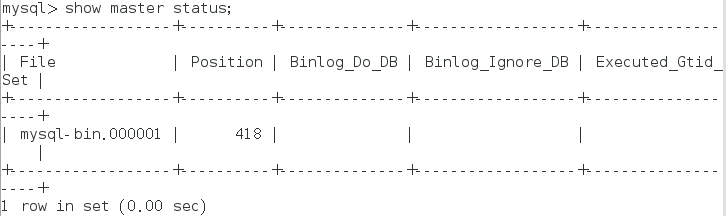


从



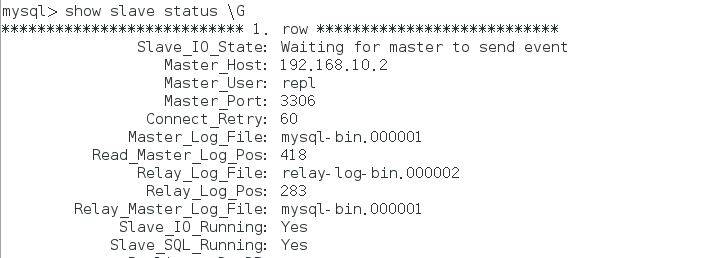
主节点





从





一、准备阶段 设置 ip 地址： [root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0

查看操作系统版本： [root@localhost ~]# cat /etc/redhat-release

查看内存： [root@localhost ~]# grep MemTotal /proc/meminfo

查看交换分区： [root@localhost ~]# grep SwapTotal /proc/meminfo

查看硬盘空间： [root@localhost ~]# df -hT /

修改主机名： [root@localhost ~]# vim /etc/sysconfig/network 修改： HOSTNAME=ocracle.benet.com

设置域名解析： [root@localhost ~]# vim /etc/hosts 添加： 192.168.1.1 oracle.benet.com

调整内核参数： [root@ocracle ~]# vim /etc/sysctl.conf 添加： fs.aio-max-nr = 1048576 fs.file-max = 6815744 kernel.shmall = 2097152 kernel.shmmax = 536870912 kernel.shmmni = 4096 kernel.sem = 250 32000 100 128 net.ipv4.ip\_local\_port\_range = 9000 65500 net.core.rmem\_default = 262144 net.core.rmem\_max = 4194304 net.core.wmem\_default = 262144 net.core.wmem\_max = 1048586

[root@ocracle ~]# sysctl -p

查看系统软件环境： [root@dbserver ~]# yum -y install binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel gcc gcc-c++ glibc

glibc-common glibc-devel glibc-headers kernel-headers ksh libaio libaio-devel libgcc libgomp libstdc++ libstdc++devel make numactl-devel sysstat unixODBC unixODBC-devel pdksh

注意：yum 安装时报错，请用下列办法解决： yum clean all yum makecache

[root@dbserver ~]# rpm -ihv /mnt/Packages/ksh-20120801-10.el6.x86\_64.rpm [root@dbserver ~]# rpm -ihv /mnt/Packages/libaio-devel-0.3.107-10.el6.x86\_64.rpm [root@dbserver ~]# rpm -ihv /mnt/Packages/numactl-devel-2.0.7-8.el6.x86\_64.rpm [root@dbserver ~]# rpm -ihv /mnt/Packages/unixODBC-2.2.14-12.el6\_3.x86\_64.rpm [root@dbserver ~]# rpm -ihv /mnt/Packages/unixODBC-devel-2.2.14-12.el6\_3.x86\_64.rpm

[root@ocracle ~]# rpm -e ksh （先移除） [root@ocracle ~]# rpm -ivh /mnt/Server/pdksh-5.2.14-36.el5.i386.rpm [root@ocracle ~]# rpm -ivh /mnt/Server/ksh-20100202-1.el5.i386.rpm （再安装） 注意： 1）ksh 与 pdksh 冲突 2）pdksh 软件包需要使用 RHEL 5.X 光盘来安装

解决中文乱码问题： 1）下载字体库（zysong.zip） 2）安装 [root@ocracle ~]# mkdir -p /usr/share/fonts/zh\_CN/TrueType [root@ocracle ~]# unzip zysong.zip [root@ocracle ~]# cp zysong.ttf /usr/share/fonts/zh\_CN/TrueType

Oracle 用户环境： [root@ocracle ~]# groupadd -g 55 oinstall [root@ocracle ~]# groupadd -g 56 oracle [root@ocracle ~]# useradd -g oinstall -G oracle oracle [root@ocracle ~]# passwd oracle （PW：123.com） [root@ocracle ~]# mkdir /opt/oracle [root@ocracle ~]# chown -R oracle:oinstall /opt/oracle [root@ocracle ~]# chmod -R 775 /opt/oracle

[root@ocracle ~]# vim /home/oracle/.bash\_profile 添加： umask 022 export ORACLE\_BASE=/opt/oracle export ORACLE\_HOME=$ORACLE\_BASE/product/11.2/db\_1 export ORACLE\_SID=orcl export LANG=zh\_CN.UTF-8

#export NLS\_LANG=”AMERICAN\_AMERICA”.ZHS16GBK(不可用) #export NLS\_LANG="SIMPLIFIED CHINESE\_CHINA.ZHS16GBK" export NLS\_LANG="SIMPLIFIED CHINESE\_CHINA.AL32UTF8" export PATH=$ORACLE\_HOME/bin:$PATH export DISPLAY=:0.0

Oracle 用户资源限制要求： [root@ocracle ~]# vim /etc/pam.d/login 添加： session required /lib/security/pam\_limits.so session required pam\_limits.so

[root@ocracle ~]# vim /etc/security/limits.conf 添加： oracle soft nproc 8192 oracle hard nproc 16384 oracle soft nofile 32768 oracle hard nofile 65536 oracle soft stack 10240

[root@ocracle ~]# vim /etc/profile 添加： if [ $USER = "oracle" ]; then if [ $SHELL = "/bin/ksh" ]; then ulimit -p 16384 ulimit -n 65536 else ulimit -u 16384 -n 65536 fi fi

环境准备完毕后，重启操作系统。 [root@ocracle ~]#reboot

二、安装 Oracle 1、准备 Oracle 软件 注意：这里使用的时 ISO 镜像文件 [root@ocracle ~]# mount /dev/cdrom /mnt/ [root@ocracle ~]# mkdir /database [root@ocracle ~]# cp -r /mnt/\* /database/

2、安装 Oracle [root@ocracle ~]# xhost +

[root@ocracle ~]# su - oracle

[oracle@ocracle ~]$ cd /database/ [oracle@ocracle database]$ ./runInstaller

密码:Com12345（密码不能以数字开头，否则后面的安装会报错。 ）

注意：修改路径为/opt/oracle/oraInventory

如果忘记解锁用户 SCOTT，怎么办？ [oracle@oracle ~]$ sqlplus sys AS SYSDB（连续回车连次，空密码登录） SQL> alter user scott account unlock； SQL> alter user scott identified by Com12345；

在完成安装后，执行玩两个脚本后，再点击“确定”。 注意：要以 root 身份登录数据库。 [root@oracle ~]# /opt/oracle/oraInventory/orainstRoot.sh //更改/opt/oracle 权限 [root@oracle ~]# /opt/oracle/product/11.2/db\_1/root.sh //调整/opt/cracle 的 root 权限

完成安装

验证： 1、使用浏览器访问 https://oracle.benet.com:1158/em/。 用户：sys 密码：Com12345 类型：SYSORACLE

2、使用命令登录数据库 [oracle@oracle ~]$ sqlplus sys/Com12345 AS sysdba

错误排查： 1、

2、

3、查询字符集

SQL> select userenv('language') from dual;

**复制**

解决单节点故障问题,需要使用复制功能,将一台服务器上的内容放置到一台或者多台服务器上的过程,称为复制

建立复制

mastr主 slave从

一个主节点只能有一个主节点,但是一个主节点不可以有多个从节点,且数据的流向是单向的(有由主到从)

1.在redis-server启动服务时,加--slaveof masterhost masterport

2.在redis-cli命令行中加入slaveof masterhost masterport

3.在配置文件中加入slaveof masterhost masterport

查看复制信息

info replication

断开主从复制的方式

1.直接结束从服务器进程

2.通过命令结束(slaveof no noe)

断开主从复制的流程

1.断开主从复制关系

2.从服务器省委主服务器(通过日志就能看出master mode enabled)

3.如果切换主服务器将会刷新旧数据

4.复制新主服务器的内容

如果要切换主服务器,先将从服务器的内容进行备份,或者让新上线的主服务器连接到从服务器获取到所有内容将连接断开重新复制(生产环境中一定要备份)

临时设置密码

给reids设置密码(告知从服务器连接的密码)

config set requirepass 123.com

解锁

auth 123.com

告知从服务器连接的密码从服务器验证config set masterauth 123.com

配置文件重写

config rewrite

只读

在从服务器中由于数据流向问题(单行)造成从服务器为只读状态,默认开启

可以通过配置文件中slave-read-only选项进行开启或者关闭,不推荐关闭

repl-disable-tcp-nodelay no 用于是否关闭tcp\_nodelay功能,默认关闭

1.当关闭时,主节点的所有内容(命令)无论大小都会及时发送给从节点,从节点获取主机点的数据,宽带消耗增加,主从延迟变小,使用于网络环境良好,或者同机房部署的情况

2.当开启时,主节点会合并较小的tcp数据包,用于节省带宽,但主从延迟增加,适用于网络环境较差,或者跨主机部署的情况

拓扑结构

1 一主一从结构

可以只在从服务器开启aof持久化实现整个架构高可用,既能保证包可用,又能保证数据的安全性

2一主多从结构

优点:实现读写分离,将aof持久化放置到从服务器减轻主服务器的压力

缺点:如果部署多台从服务器,主服务器将会rdb文件发送给多台从服务器造成宽带压力

1. 树状结构

优点：减轻主服务器发送rdb文件消耗带宽过多得压力，还可以在从服务器开启持久化保证数据得安全性

缺点：主节点宕机,造成整个架构崩溃

复制原理

1.保存主节点的信息

2.从节点通过每秒设定好的定时任务进行执行,准备复制,当发现主节点时从节点会尝试于主节点建立连接,从节点会常见socket套接字文件,专门用于接收注解带你的复制命令,如果从节点无法建立连接,定期任务无限循环知道成功为止或者slaveof no one 断开复制

3.连接成功,从节点发送ping命令,请求通信(检查套接字文件是否可用,检测主节点是否接收命令)如果ping命令没有收到主节点的ping,或者网络超时或者主节点阻塞,从节点断开连接,下次继续发送连接

4.执行权限验证

5.同步数据,主节点将命令全部发送给从节点

6.复制持续进行

全量复制

从节点学到主节点的所有数据

主从节点都会保留一份master\_replid数据属于复制中主服务器的id 号,id会随着主节点的id号和自己保存的id号不一致时,即使为切换主服务器将会执行全量复制

增量复制

从节点学到主节点的部分数据

当从服务器宕机再上线时,首选会通过master replid来判断主服务器是否为复制的服务器,如果id相同则判断为同一个主服务器,将会选择是否进行增量复制,当从服务器中丢失的数据能够在复制挤压缓冲区找到,那么执行增加复制,如果复制的偏移量超过了复制挤压缓冲区的长度,将会执行全量复制

从节点学到主节点部分数据

repl\_backlog\_active:1 开启复制挤压缓冲区

repl\_backlog\_size:1048576 缓冲区的大小(默认1M)

repl\_backlog\_first\_byte\_offset:1起始的偏移量

repl\_backlog\_histlen:84 已保存的有效长度

**哨兵**

为了解决主节点故障造成的从服务不能使用的情况,出现哨兵功能,哨兵属于独立的redis进程,不存在任何数据,当主节点宕机,哨兵会将从节点晋升为主节点继续提供服务,解决了整过程由人工干预的问题(故障转移)

以住两从哨兵如何进行故障转移

1.主节点发生故障,从节点连接失败,复制断开

2.如果主机点无法启动,许哟啊将其中的一个从节点晋升为主节点进行服务器

3.哨兵将会把从节点晋升为主节点,然后更新消息

4.哨兵让其他的从节点连接到晋升为主节点的服务器上

合理部署多台哨兵的优点

1.有效方式误判产生

2.部署多个哨兵可以保证集群健壮(哨兵许需要部署奇数个,且最少为3个)’

sentinel monitor mymaster 192.168.1.12 6379 2 设置监控的主节点信息,ip端口,2表示投票执行故障转移的人数

sentinel down-after-milliseconds mymaster 10000 判断主节点宕机多久执行故障转移(默认30000毫秒=30秒)

sentinel parallel-syncs mymaster 1 故障转移时,全量复制的人数

sentinel failover-timeout mymaster 180000 故障转移的超时时间

哨兵启动方式

1.redis-sentinel 哨兵配置文件

2.redis-sever –sentinel 哨兵配置文件

客观宕机

超过半数的哨兵判定宕机 +odown

主观宕机

当个哨兵判断宕机 +sdown